.NET PROGRAMMING

SQL PROGRAMMING AND DBMS

DATABASE DEFINITION LANGUAGE

Chia Yuen Kwan

isscyk@nus.edu.sg

Objectives

- By the end of this lesson, students should be able to use SQL command for
  - Data Definition Language to:
    - Create Tables with
      - Primary Keys
      - Foreign Keys
    - Create Indexes on Tables
    - Alter Table
    - Drop Table and Indexes
  - Define Constraints
    - Required Data Constraint
    - Validity Constraint
    - Entity Integrity
    - Referential Integrity

Topics

- **DDL (Data Definition Language)**
  - **Create / Alter / Drop**
- Defining Constraints

Data Definition Language

- DDL (Data definition language) portion of SQL
  - define and manage the structure and organization of the stored data (ie objects) and relationships among the stored data items.
  - Objects includes Tables, Views, indexes, etc
- Commands for DDL include:
  - CREATE : Create a new object in the Server system.
  - DROP : Remove the object from the system.
  - ALTER : Change the characteristic of the objects in the Server that has been present in the system.

CREATE Statement

- Function of a CREATE Statement:
  - Define new objects (database, table, index, views, etc)
    - CREATE TABLE
    - CREATE INDEX
    - CREATE VIEW

CREATE TABLE

- Example:

records created must have value in the column

```
CREATE TABLE GoodCustomers
(CustomerID          nvarchar(4)          not null,
 CustomerName        nvarchar(50)         not null,
 Address             nvarchar(65)         not null,
 PhoneNumber         nvarchar(9),
 MemberCategory      nvarchar(2)          not null,
 PRIMARY KEY (CustomerID,MemberCategory))
```

composite primary key

- Interpretation:
  - Creates a table
    - named GoodCustomers with five columns: CustomerID, CustomerName, Address, PhoneNumber and MemberCategory,
    - With PhoneNumber can have Null values
    - having a Composite Primary Key consisting of CustomerID and MemberCategory.

CREATE TABLE

- Example (with foreign key definition) :

```
CREATE TABLE ProducerWebSite
(Producer                    varchar(50)   not null,
 WebSite                     varchar(200)  not null,

 PRIMARY KEY(Producer),
 FOREIGN KEY (Producer)      REFERENCES
                             Producers (Producer))
```

table-name (column-name)

- Interpretation:
  - Creates a table
    - named ProducerWebSite with two columns: Producer and Website,
    - having a Producer Column as the Primary Key
    - and having Producer Column of Producers Table as the Foreign Key

CREATE TABLE

- Alternative syntax (indicating constraint name) :

```
CREATE TABLE ProducerWebSite
(Producer                    varchar(50)   not null,
 WebSite                     varchar(200)  not null,

 PRIMARY KEY(Producer),
 CONSTRAINT ProducerWS_FK_1 FOREIGN KEY (Producer)
                            REFERENCES Producers
                            (Producer))
```

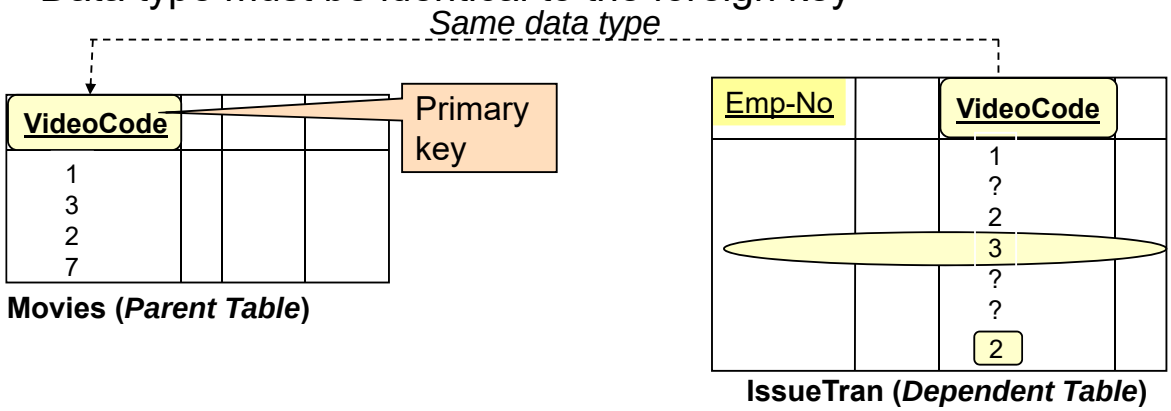
foreign key column

Constraint name

- Interpretation:
  - Creates a table
    - named ProducerWebSite with two columns: Producer and Website,
    - having a Producer Column as the Primary Key
    - and having Producer Column of Producers Table as the Foreign Key

Create Table – Foreign Key

- Associated column in the parent table
  - must be a primary key (or unique index)
  - Data type must be identical to the foreign key



- Rows can be inserted or foreign key column can be updated in the dependent table only if
  - (1) there is a corresponding primary key value in the parent table, or
  - (2) the foreign key value is set null.

CREATE INDEX

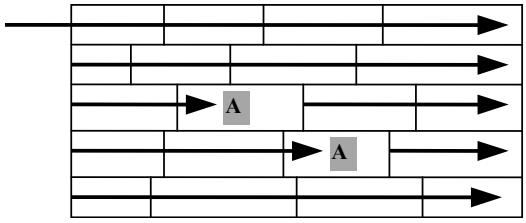
```
CREATE [UNIQUE] INDEX index-name
       ON table-name (column-name [,......] [ASC|DESC] )
```

index-name:	Unique name that identifies the index
table-name:	Name of table being indexed
column-name:	Name of column(s) on which index is created. A limit is often imposed on the number of columns that can be used in a compound index

{ } / [ ] denotes optional items

Searching without Index Key

- Searching a record without index key (or primary key) involves scanning the whole table
- *E.g. Select * from Customers*
  where membercategory = 'A'



Records in Customer table are stored in random order

(if the column membercategory is not defined as primary or index key, the above query will result in table scan by the DBMS)

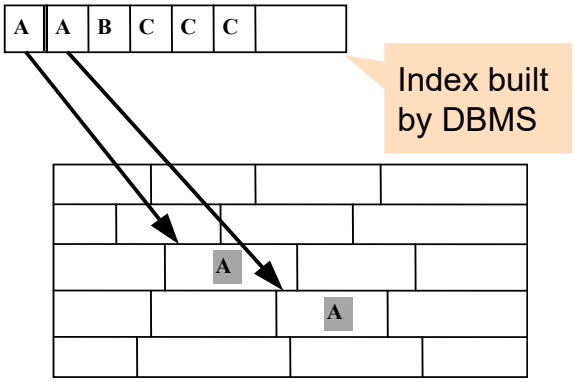
Index Key

- Index Key
  - optimise searching

An index is a listing of keys stored in order, accompanied by location to the record.

Searching a record using an index key can be speed up.

Table Read using matching index.



Index built by DBMS

*Select * from Customers*
where membercategory = 'A'
(membercategory is defined as an index key)

CREATE INDEX

- Example:

index name

```
CREATE UNIQUE INDEX gdCust_idx ON
GoodCustomers(PhoneNumber)


CREATE UNIQUE INDEX gdCust_idx ON
       GoodCustomers(CustomerID,CustomerName)
```

Table-name (column-name)

```
CREATE INDEX Cust_idx ON Customers(Address)
```


Index Key

- An index key can be unique or non-unique
- A primary key is actually a primary unique index

Disadvantages of indexes

- Every index increases the storage space in the database
- When data are inserted, updated or deleted, the index must be updated. An index saves time in retrieval of data, but it costs time in insert, update or delete operation

DROP Statement

- Function of a DROP statement:
  - *Remove (erase) an existing object that is no longer needed*

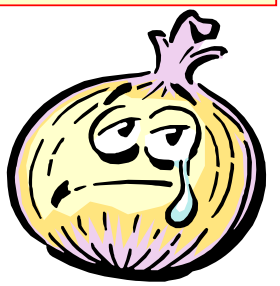
DROP Command

```
DROP [table-name | index-name | view-name ]
```

DROP Statement

- Example:

```
DROP TABLE GoodCustomers
DROP INDEX Cust_idx ON GoodCustomers
```



ALTER Statement

- Function:
  - *Change the definition of an existing table.*

```
ALTER TABLE table-name { option(s) }
    {ADD column-name  data-type  {NOT NULL} {WITH DEFAULT},
    |DROP column-name [ , ....]
    |ALTER COLUMN column-name column-type
    |ADD UNIQUE (column-list)
    |ADD PRIMARY KEY key-name (column-list)
    |ADD FOREIGN KEY (column-list) REFERENCES table-name (column-
        name)
       [ON DELETE {CASCADE | NO ACTION} ]
    |DROP PRIMARY KEY
    |DROP FOREIGN KEY constraint-name ]
    |DROP CHECK }
```

| denotes one and only one of the command is to be selected and not all

ALTER Statement

- Example:
  - Adding a Column

```
ALTER TABLE GoodCustomers
ADD CustomerPassword nvarchar(25)
```

  - Dropping a Column

```
ALTER TABLE GoodCustomers
DROP COLUMN CustomerPassword
```

  - Dropping a Primary Key

```
ALTER TABLE GoodCustomers
DROP PK_GoodCustomers
```

Primary key constraint name

ALTER Statement

- Example:
  - Adding Primary Key

```
ALTER TABLE Country ADD PRIMARY KEY (CountryCode)
```

  - Adding Unique Key

```
ALTER TABLE IssueTran ADD UNIQUE (TransactionID)
```

  - Adding Foreign Key

```
ALTER TABLE IssueTran
ADD FOREIGN KEY (CustomerId) REFERENCES Customers (CustomerId)
```

If you encounter problem in the process of adding foreign key, check whether existing data fulfills referential integrity constraints

Topics

- DDL (Data Definition Language)
  - Create / Alter / Drop
- **Defining Constraints**

SQL for Data Integrity

- Data integrity can be lost in many ways:
  - Invalid data added to data base
  - Existing data modified to a incorrect value
- SQL can be used to enforce date integrity by inserting the following types of constraints when the object is created (using DDL):
  - Required Data
  - Validity Checking
  - Entity Integrity
  - Referential Integrity

Required Data Constraints

- Required Data
  - Fields (or columns) cannot accept null value
  - Usually handled by **Not Null**
  - Eg.:

records does not accept record with no value in Producer column

```
CREATE TABLE ProducerWebSite
(Producer                    nvarchar(50)  NOT NULL,
 WebSite                     nvarchar(200) NOT NULL
 PRIMARY KEY(Producer)
 FOREIGN KEY(Producer)       REFERENCES Producers)
```


Validity Checking Constraint

- Validity Checking
  - Columns having a particular range or format

```
CREATE TABLE StockAdjustment
(VideoCode          SmallInt            not null,
AdjustmentQty       Int,
DateAdjusted        DateTime,
WhoAdjust           nvarchar(20),
AdjustReason        nvarchar(50),
CONSTRAINT Con_VideoCode CHECK(VideoCode BETWEEN 0 AND 99999))
```

The table can only accept video code that falls within a certain range of values

Entity Integrity Constraint

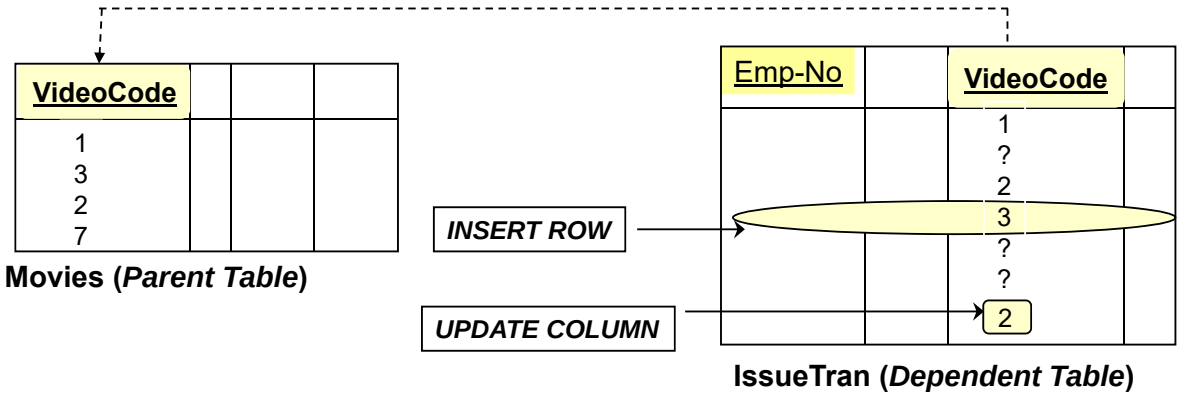
- Entity Integrity (or Entity Constraint)
  - Each row in the table to have a unique value for a particular column(s)
  - Usually implemented using a **UNIQUE** constraint or a **PRIMARY KEY** constraint
  - Eg.:

defines a unique index key

```
CREATE TABLE Producers
(Producer            nvarchar(50)  not null,
 ProducerName        nvarchar(50)  not null      UNIQUE,
 CountryCode         nvarchar(3)   not null,
 PRIMARY KEY(Producer,ProducerName),
 FOREIGN KEY(CountryCode)REFERENCES Country(CountryCode)
                                          ON DELETE CASCADE)
```


Referential Integrity Constraint

- Enforced through Foreign Key:
  - *Every non-null value in a foreign key must have a corresponding value in the primary key which it references.*

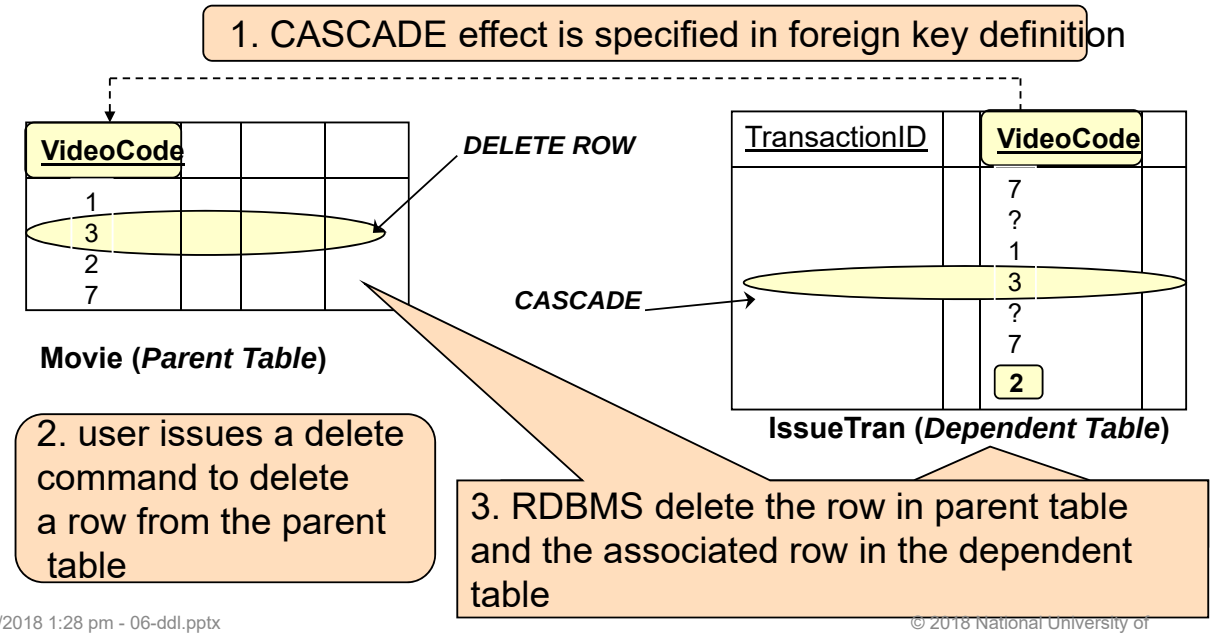


A row can be inserted or a column updated in the dependent table only if (1) there is a corresponding primary key value in the parent table, or (2) the foreign key value is set null.

Referential Integrity – Effects for Deletes Operation (1)

Database designers may explicitly declare the effect (e.g. CASCADE) if a row is deleted from the **parent** table on the dependent table.

- CASCADE deletes associated dependent rows if parent table's row is deleted



SQL for Referential Integrity

- SQL data definition for defining referential integrity constraints:

Parent table:

```
CREATE TABLE Movies
(VideoCode              smallint              not null,
… other column definitions
PRIMARY KEY (VideoCode) )
```

- Dependent table:

```
CREATE TABLE IssueTran
(TransactionID        smallint              not null,
 VideoCode            smallint              not null,
 … other column definitions
 PRIMARY KEY(TransactionID),
 FOREIGN KEY(VideoCode)  REFERENCES Movies(VideoCode)
                                  ON DELETE CASCADE)
```

SQL for Referential Integrity

- Interpretation of the commands :
  - The above example basically creates a Movie table with VideoCode being the primary key
  - The IssueTran table is created next with a foreign key, VideoCode.
  - Note that the IssueTran table references to the Movies table with a delete cascade referential integrity
  - In other words, if a row in Movies table is deleted (ie. the video code no longer exists), every transaction row in the IssueTran table having the VideoCode will be deleted.
- Defining referential integrity rules using SQL DDL is known as Declarative Referential Integrity.
- Enables enforcement at the database server level, eliminating the possibility of application errors.

Referential Integrity – Effects for Deletes Operation (2)

SQLServer:

NO ACTION (default): raise error (operation on parent row not allowed) if there exists at lease one row in the referencing table

CASCADE: deletes associated dependent rows if parent table's row is deleted

The above effects can be defined for Update and Delete operations.

Summary

- Data Definition Language:
  - Create Tables with
    - Primary Keys
    - Foreign Keys
  - Create Indexes on Tables
  - Alter Table
  - Drop Table and Indexes
- Define Constraints
  - Required Data Constraint
  - Validity Constraint
  - Entity Integrity
  - Referential Integrity